Coywin implementation notes

Basically just writing this down so i remember why things are set up this way.

Pow loop:

Just doing a basic while True sha256 loop until hash.startswith("0000").

no scrypt or memory hard stuff right now. its just to slow down the python script so it doesnt generate 1000 images a second and crash the node. obviously any gpu miner will destroy this target instantly so we need to fix the retargeting later.

PQC signatures:

Using dilithium5 (ml-dsa-87) from the pqcrypto package, could have just used ecdsa but wanted to mess with post-quantum stuff.

the signatures are huge so instead of stuffing them in the png exif (which twitter strips anyway) it just dumps a detached .pqc file next to the image.

LSB watermark:

The EXIF stripping issue is why the lsb steganography exists.

coywin_watermark.py takes the image width*height and runs sieve of eratosthenes to get prime numbers.

Those primes are used as the 1d array indices to embed the payload bits into the rgb channels. payload is just "coywin_" + id xor'd with a sha256 keystream of the private key.

note: this is super fragile. if someone *resizes the image or compreses it to jpeg, the rgb values change and the prime coordinates shift. the extraction will just spit out garbage. only works on the raw untouched pngs.

BIP-Coywin addressing:

Hex addresses look awful so bip_coywin.py grabs hex_hash[26:38] (12 hex chars).

splits it into 4 chunks of 3 chars.

3 hex chars = 12 bits = 0 to 4095.

does modulo 4096 on that number to pull a syllable from Kamus_Suku_Kata_12bit_4096_R_Winjay_Final.csv.

String them together and you get a 15ish letter alias. collisions will probably happen eventually since the dictionary is small but whatever it works for now.

Building the exe:

Using pyinstaller on miner.py makes a massive 40mb+ exe because it pulls in numpy and the pqcrypto c binaries. windows defender keeps flagging it too. might need to look into nuitka later.